Journal of Computer Science 4 (11): 934-941, 2008
ISSN 1549-3636

Embedding Malaysian House Red Ant Behavior into an Ant Colony System

Zulaiha Ali Othman, Helmi Md Rais and Abdul Razak Hamdan
Faculty of Information Science and Technology, Centre of Artificial Intelligent Technology,
Universiti Kebangsaan Malaysia, Malaysia

Abstract: Problem statement: Ant Colony System (ACS) is the most popular algorithm used to find a shortest path solution in Traveling Salesman Problem (TSP). Several ACS versions have been proposed which aim to achieve an optimum solution by adjusting pheromone levels. However, it still has a room on an improvement. This research aims to improve the algorithm by embedding individual Malaysian House Red Ant behavior into ACS. **Approach:** Modeling individual ants' ability reconstructing a path can provide a general idea on how such behavior can improve existing basic ACS ability in finding solution. This study presents a model of Dynamic Ant Colony System with Three Level Update (DACS3) which developed by embedding such behavior into ACS. The three level phases of pheromone updates are: local construction, local reinforcement and global reinforcement. The performance of DACS3 is measured by its shortest distance and time taken to reach the solution against several ant colony optimization algorithms (ACO) on TSP ranging from 14 to 100 cities by running the algorithm in c language. **Results:** The result shows that DACS3 has reached the shortest distance benchmark for dataset 14, 30 and 57 and has 0.5% differences for data set 100. While, others ACO manage to reach for data set 14 and 30 only and reached about 2.5% differences for data set 100. For dataset 57, DACS has reached 4.6% differences whilst ACS has reached 2.5% differences. **Conclusion:** Embedding a simple behavior of a single ant into ACS influences an achievement to reach an optimal distance and also can perform considerably faster compare to other ACO's algorithms.

Key words: Dynamic Ant Colony System (DACS), Traveling Salesmen Problem (TSP), optimization, swarm intelligent

INTRODUCTION

Today's business environment is increasingly complex and dynamic, with substantial flexibility required in operations. This is especially true of the logistics and transportation industry, where the need to deliver on time and to fulfill changes in customer requests makes it important to find the shortest paths for a delivery route. People, as we know, have a reduced ability to see the overall problem, particularly when the problem is relatively complex in term of its size and constraints. However, this inability can be overcome with the help of certain advanced optimization methods, which can aid humans in expediting the process.

The Ant Colony System (ACS) is one the most successful algorithm used in combinatorial optimization problems such as the Traveling Salesman Problem (TSP), Vehicle Routing Problem (VRP), Job-Shop Scheduling Problem (JSP) and Quadratic Assignment Problem (QAP)[1,6]. The algorithm is inspired by the foraging behavior of a colony of ants, which communicate through chemical substances called pheromones, acting as a memory preservation mechanism and providing guidance for ants in searching for shortest paths[2].

The principle of cooperation is the backbone of these algorithms. However, observing the behavior of a single ant can add value to the principle. Manipulating pheromone substances is a simple addition that can help to find the best solution. Therefore, many versions of ACS algorithms have been produced to find the shortest path by using the principle of cooperation among the ants. This study looks at individual ants' behavior in trying to reconnect paths previously laid by the colony when an obstacle is placed on such paths. Such blockages add another level of pheromone updates, which could contribute to faster optimum solutions.

Corresponding Author: Zulaiha Ali Othman, Faculty of Information Science and Technology, Centre of Artificial Intelligent Technology, Universiti Kebangsaan Malaysia, Malaysia Tel: 03 89216754 Fax: 03 89216184

This study concentrates on an improvement of an ACS applied to the TSP domain, as first presented by Marco Dorigo[3]. The problem is to find the shortest tour of all the cities, where all cities are connected to each other and visiting each city only once.

MATERIALS AND METHODS

Ant behaviors: Nature is a good source of solutions to problems faced by humans. Ants provide a good example for the case of transporting goods or finding shortest paths. Ants are social insects which cooperate through group communication, laying down chemical substances called pheromones[4] to mark locations that have already been visited. The pheromones also serve as a reference for the return route back to their nest. These pheromones are then used by other ants as an indicator of the best path between the nest and food sources. The amount of pheromone laid determines whether the path is desirable to be taken by others; higher pheromone levels indicate more desirable routes.

Research on social insects began in the early twentieth century, when the South African scientist Eugene Marais (1872-1936) focused his attention on the behavior of termites, which he refers to as White Ants in his research. His research was then picked up by Konrad Lorenz (1903-1989) in his studies of imprinting and instinctive behavior of termites. Although both scientists are considered to be pioneers in the field of Ethology, the scientific studies of animal behavior, they were unaware of the actual mechanics of termite communication[5]. The answer to this question was discovered in the 1940s and 1950s when a French biologist named Pierre Paul Grasse investigated how termites communicate. He discovered that social insects react to significant stimuli that activate genetically encoded reactions called stigmergy, which describe as a type of indirect communication that workers stimulated by the performance they achieved[6]. The characteristics of stigmergy were also found in single and double bridge experiments on Argentine Ants, where they studied the pheromones laying and following behavior of ants[7].

Margo Dorigo *et al*.[11] applied the results of these experiments to artificial ants, basing his research on ethologist studies that found ants established shortest paths based on pheromone trails. He took it one step further to study the random movement of ants, which he referred to as autocatalytic behavior, where ants move at random, detect an existing trail, decide to follow and then re-enforce by laying down its own pheromones. Autocatalytic behavior is a process of positive and negative feedback, or pheromone reinforcement and evaporation, that causes very rapid convergence[3,8]. Figure 1 show the experimental setup of Marco Dorigo, where an obstacle was placed in the path of multiple ants.

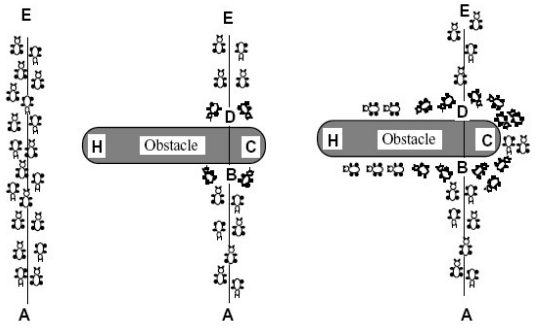

Fig. 1: Obstacle experiment

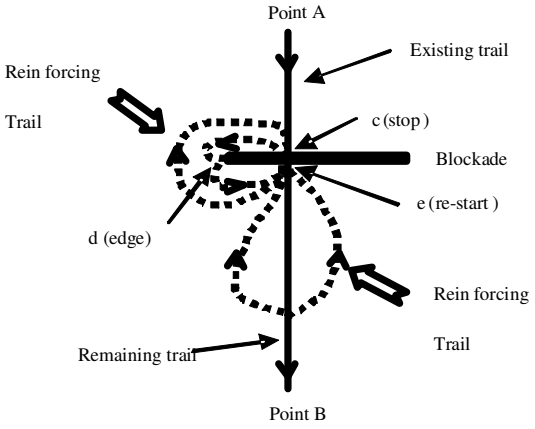

Fig. 2: Single ant obstacle experiment

The principle of cooperation among swarm insects is that communication among individuals contributes to the survival of the group as a whole[9]. We have conducted experiments to several colonies of Malaysian House Red Ants by using the experimental method of Marco Dorigo. An object or obstacle is placed on a single ant's normal path to find out how it behaves[10]. Malaysian House Red Ant is a small size red ant species and because of its size, it walks pattern is very slow and can easily be observed. The experiment is conducted at a time when there are not many ants in the colony or the colony is inactive, normally late at night. It is very important to find a good trail set up by the early colony activities in order for an individual ant to walk with. Figure 2 shows ant behavior in constructing its paths.

An ant travels along its normal path, Point A → Point B, following the strategic rules set out by the pheromones. When we put an obstacle in its path, the ant begins to search for an alternative route. It begins by examining both edges of the obstacle several times. Once it chooses a shorter edge to continue along, it begins to set up the path by visiting the shorter edge

point d from point c and then tries to find a point e which returns to the existing trail. Once it does so, reinforces the newly constructed path by revisiting (c,d,e) several times.

Once the new alternative path has received enough reinforcement, the ant then continues its journey using the path that existed before the obstruction. However, after some distance, it returns to the point c, where it restarts its journey and continues to reinforce the remaining path several times.

From our observation, we can see that many of the individual ant experience a chaotic behavior[19] but few has a bit of an intelligent try to reconstruct the trail. Thus, by observing these few intelligent ants, we can conclude that to construct paths or a tour, there are three events involved: one event for path construction and two events for trail reinforcement.

Ant Colony Optimization (ACO) metaheauristic background:

Ant System (AS): AS was first introduced and applied to TSP by[3]. Initially, ants were placed on n cities, moving from city r to city s using probabilistic formula called random-proportional rules:

$$S=\begin{cases}\dfrac{[\tau(r,s)].[\eta(r,s)]^{\beta}}{\sum_{u\in j_k(e)}[\tau(r,u)].[\eta(r,u)]^{\beta}} & \text{if } s\in J_k(r)\\ 0 & \text{otherwise}\end{cases} \quad (1)$$

The Euclidean distance formula ($d_{ij} = ([x_i-x_j]^2+[y_i-y_j]^2)^{1/2}$) is used to calculate the move distance between city r and city s. $\tau(r,s)$ is the trail intensity of edge (i,j) and the visibility $\eta(r,s) = 1/d_{ij}$ is the inverse distance of move (r,s). β is a parameter which determines the relative importance of distance versus pheromone level ($\beta>0$). $J_k(r)$ is the set of cities that remains to be visited by ant k when positioned at city r. After all ants have completed a tour, the pheromone level on all edges is updated using a local pheromone updating rule[6], as follows:

$$\tau(r,s)\leftarrow(1-p).\tau(r,s)+\sum_{k=1}^{m}\Delta\tau_k(r,s)$$

$$\text{wher}\quad \Delta\tau_k(r,s)=\begin{cases}1/L & \text{if } (r,s)\in \text{tour done by ant k}\\ 0 & \text{Otherwise}\end{cases} \quad (2)$$

(1-p) is the pheromone decay parameter ($0<p<1$), representing the trail evaporation when the ant chooses a city and decide to move. L_k is the length of the tour performed by ant k and m is the number of ants.

Ant Colony System (ACS): Luca Maria Gambardella and his colleague[11] have modified the AS algorithm, introducing the ACS to provide more balance and guidance in searching. Firstly, the state transition rules (pseudo-random proportional rules) combine Eq. 1 and 3, providing a direct method to balance between exploring new edges and exploiting existing edges. Secondly, only edges that belong to the best ant tour are allowed to undergo pheromone updates through a global pheromone updating rule. Finally, the local pheromone updating rule is applied while ants are trying to construct a solution[12]:

$$S=\begin{cases}\arg\max_{u\in J_k(r)}[\tau(r,u)].[\eta(r,u)]^{\beta} & \text{if } q\le q_0 \text{ (Exploitation)}\\ S & \text{Otherwise (Biased exploration)}\end{cases} \quad (3)$$

Each ant builds a tour by repeatedly applying the state transition rules. q is a random number uniformly distributed [0.1] and q_0 is the parameter ($0\le q_0\le 1$) which determines the relative importance of exploitation versus exploration.

While constructing a tour, an ant will modify the pheromone level on the visited edges using the local pheromone updating rule Eq. 4. L_{mn} is the tour length produced by the nearest neighbor heuristic and n is the total number of cities[13]:

$$\tau(r,s)\leftarrow(1-p).\tau(r,s)+p.\Delta\tau_0$$

$$\text{wher}\,\Delta\tau_0=(1/L_{mn}.n) \quad (4)$$

After all ants have constructed a tour, only the globally best ant which produces the shortest tour from the beginning of the trial will be allowed to do pheromone updates using the global pheromone updating rule Eq. 5. L_{gb} is the length of the globally best tour from the beginning of the trial[14]:

$$\tau(r,s)\leftarrow(1-p).\tau(r,s)+p.\Delta\tau(r,s)$$

$$\text{wher}\,\Delta\tau_k(r,s)=\begin{cases}(L_{gb}) & \text{if } (r,s)\in \text{Global best tour}\\ 0 & \text{Otherwise}\end{cases} \quad (5)$$

Max-Min Ant System (MMAS): Stutzle and Hoos[15] considered ACS when developing MMAS, but otherwise make improvements directly onto AS. MMAS is different in three ways. The first improvement is that only one ant would update the pheromone, which is the same as the model of ACS, but

it could choose whether to update one solution of the current iteration or follow the global best solution. Second, the pheromone strength was to be bounded to upper and lower limit[t_{max}, t_{min}] in order to avoid search stagnation. Lastly, the initial value for pheromone strength was initializing to t_{max} which was intended to provide a higher search exploration of solution at the beginning of the algorithm runs[16]:

$$\tau(r,s) = p.\tau(r,s) + \Delta\tau_{rs}^{best}$$
$$\text{wher}\,\Delta\tau_{rs}^{best} = 1/f(s^{best}) \quad (6)$$

Equation 6 shows how MMAS performs the pheromone updates. s^{best} could be either the global solution from the beginning of the trail or the solution of the current iteration. However, MMAS prefers to use the iteration's best solution, this is the most important element in the MMAS search. By making this choice, the solution elements that frequently occur would receive large pheromone reinforcements which result in considerably different solutions from iteration to iteration:

$$\tau(r,s)^* = \tau(r,s) + \delta.\left(\tau_{max} - \tau(r,s)\right) \text{with } 0 < \delta < 1$$
$$\text{where } \tau_{max} = \frac{1}{1-p}.\frac{1}{f(S^{gb})} \quad (7)$$

One stabilizer mechanism that was introduced in MMAS is Pheromone Trails Smoothing (PTS). PTS, represented by Eq. 7, was used in MMAS to improve performance. When MMAS is close to convergence, this mechanism helps increases pheromone trails proportionally to the maximum pheromone trail limit. The advantage of the mechanism is that the information gathered during the run is not completely lost, but merely weakened. This mechanism is interesting when a long run is allowed.

Dynamic Ant Colony System (DACS): Yi and Gong[17] have proposed an algorithm which is a direct improvement of AS, but considers the improvement made in ACS and MMAS by introducing dynamic decay parameter (1-p[τ(r,s)]) to avoid pheromone levels growing too high and reaching local optima. With the theory that intense pheromones evaporate more quickly than faint pheromones, the dynamic decay parameter was applied to both the local pheromone updating rule, Eq. 8 and the global pheromone updating rule, Eq. 9:

$$\tau(r,s) \leftarrow \left(1 - p\left[\tau(r,s)\right]\right).\tau(r,s) + p.\Delta\tau_0$$
$$\text{where} \quad \Delta\tau_0 = \left(1/L_{mn}.n\right) \quad (8)$$

Yi and Gong also try to accelerate the computation of the solution by allowing the best and worst tour done by ants to do pheromone updates. L_{gw} is the length of the globally worst tour and L_{gb} is the length of the globally best tour:

$$\tau(r,s) \leftarrow \left(1 - \left[p.\tau(r,s)\right]\right).\tau(r,s) + C$$
where
$$C = \begin{cases} p.\Delta\tau(r,s) & \text{if } (r,s) \in \text{Global Best Tour} \\ -p.\Delta\tau(r,s) & \text{if } (r,s) \in \text{Global Worst Tour} \\ 0 & \text{if } (r,s) \in \text{Others} \end{cases} \quad (9)$$
and
$$\Delta(r,s) = \begin{cases} (L_{gb})^{-1} \\ (L_{gw})^{-1} \end{cases}$$

These ACO algorithms adopt several other strategies[18] to improve solution quality and/or achieve better performance, such as candidate list, don't look bit and tour improvement heuristics.

Dynamic ant colony system 3 level updates (DACS3): DACS3 considers the basic concepts introduced in ACS and DACS. However, we apply individual ant behavior as shown in Fig. 2, so DACS3 differs from previous systems in three ways. First, capturing all knowledge from the group and updating the pheromone level once the knowledge becomes available would expedite the process and increase the chances of finding a better solution. Second, a dynamic penalty on worst tours would open up chances for ants to navigate, limiting intentions and providing caution in an ant's decision to move. Finally, only one best tour from group performance is considered when applying the global pheromone updating rule, which is compared with two subdivided sections (best of the best and worst of the best). Figure 3 shows the workflow of the DACS3 model.

In the local construction phase, all cities are considered as starting cities r. Every ant would have to make a complete tour and the decision to choose which city s to move to is provided by state transition rules Eq. 1 and 3. Even though we use both exploitation and exploration decision rules to move, we favor exploration as the main strategy in finding solution. Every time an ant visits a city s, it modifies the pheromone level by using the local pheromone updating rule, Eq. 4. The reason why we used the ACS version of local pheromone updating rule is because we believed that when an ant moves to a new location, it would make a constant pheromone deposit and evaporation.

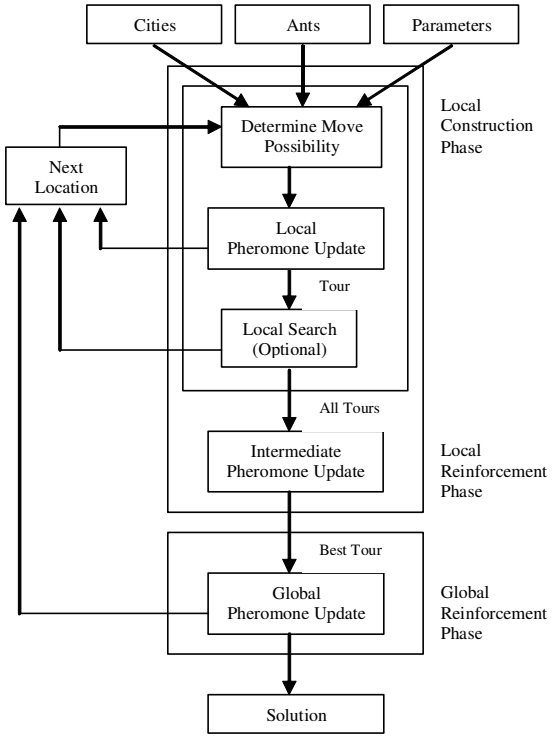

Fig 3: DACS3 Diagram

Once all ants in the group completed their tours, the available knowledge of every member of the group will then be used to modify the pheromone level using the intermediate pheromone updating rule Eq. 10 in the local reinforcement phase. The updates are necessary before all ants in the group can be given a new task to complete. In this phase, the dynamic decay parameter will be used because it helps to alleviate an early stagnation, reducing the possibility of pheromone levels growing too high.

All current completed tours in the group will be compared to the group best tour in the current iteration and to the group worst tour from the beginning of trial. If no match is found, the edges would experience normal dynamic evaporation. This method will boost very effort the ants make to produce the best tour but:

$$\tau(r,s) \leftarrow (1-[p.\tau(r,s)]).\tau(r,s) + \Delta C \quad (10)$$

where

$$\Delta C = \begin{cases} p.\Delta\tau(r,s) & \text{if } (r,s) \in \text{Group Best Tour} \\ -[p.\tau(r,s)].\Delta\tau(r,s) & \text{if } (r,s) \in \text{Group Worst Tour} \\ 0 & \text{if } (r,s) \in \text{Others} \end{cases}$$

and

$$\Delta\tau(r,s) = \begin{cases} (L_{grb})^{-1} \\ (L_{grw})^{-1} \end{cases}$$

dampen the worst tour from group performances. The dynamic penalty[$p.\tau(r,s)$] is used to caution all ants off the bad paths on the next tour. L_{grb} is the total distance of the best tour in the current iteration and L_{grw} is the total distance of the worst tour of the group from the beginning of the trial.

The pheromone level is again modified using the global pheromone updating rule, Eq. 11. Only the best tour from the group performance will be considered for the pheromone updates. Every move in the solution or the tour will be compared with every move in the complete tours gathered for two categories, best of the best and worst of the best. This method will provide better search guidance in the effort to search for a better solution:

$$\tau(r,s) \leftarrow \left(1-\left[p.\tau(r,s)\right]\right).\tau(r,s) + \Delta C$$

where

$$\Delta C = \begin{cases} p.\Delta\tau(r,s) & \text{if } (r,s) \in \text{Global Best Tour} \\ -\left[p.\tau(r,s)\right].\Delta\tau(r,s) & \text{if } (r,s) \in \text{Global Worst Tour} \\ 0 & \text{if } (r,s) \in \text{Others} \end{cases} \quad (11)$$

and

$$\Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1} \\ (L_{gw})^{-1} \end{cases}$$

L_{gb} is the total distance of the globally best tour (best of the best) and L_{gw} is the total distance of the globally worst tour (worst of the best) from the beginning of the trial. Fig. 4 shows how the DACS3 algorithm works. In this algorithm, we do not apply any additional strategic techniques or heuristics.

The experimental setup: The algorithm was tested using several datasets taken from TSPLIB. The algorithm was developed in the C language in Microsoft Visual Studio. Testing was performed on a machine with an Intel(R) Pentium (R) M 1.86GHz processor with 1 gigabyte of physical memory. Burma14 (GEO), Oliver30, Berlin52 and KroC100 (Euclidean) were taken as a case study for algorithm comparison.

The experiments sought to determine which algorithms could reach optimal distance, if all tested algorithms were able to find it and then performance speed would be the second measurement. For comparison, the first column is the best distance from the beginning of the trail, as compared to the benchmark distance. The second column shows the number of iteration required to come up with the best distance. The third column is an average time from 15 trials. Distance was measured by the integer distance

(the roundup distance from each moves) and the real distance (in the bracket). The value is measured in Euclidean and GEO distances. Real distance was used as a measurement in calculating distances for Euclidean datasets, while integer distance was used as the basis of the distance calculation for GEO datasets. Table 1 shows the parameter settings for the DACS3 experiment, which is a similar setup from research on the ACS and DACS algorithms[2,5,10,13,16].

```
GlobalBestTour = ∞ ;
GlobalWorstTour = 0;
LocalGroupWorstTour = 0;
Initialize pheromone level for all cities = τ0 ;
Generate initial solution using Nearest Neighbor (NN) heuristic;
CPU timer starts;
/* Trial begins */
Do
   /* Iteration begins */
   If i <= n
      LocalGroupBestTour = ∞ ;
      For k = 1 to m
         Start city = i;
         Do
            Select the next city j;
    /*Perform Local Pheromone Update*/
    Update trail level τij ; (Equation (4))
         While (until all cities visited)
      EndFor
      /*Perform Intermediate Pheromone Update*/
      For k = 1 to m
         Compute tour distance;
         If (tour distance < LocalGroupBestTour)
            LocalGroupBest = current solution;
         Else if (tour distance > LocalGroupWorstTour)
    LocalGroupWorst = current solution;
         Else
    /*Pheromone updates for others*/
    Update trail level τij ; (Equation (10))
         EndIf
      /*Pheromone updates for LocalGroupBest &          LocalGroupWorst*/
      Update trail level τij  for LocalGroupBest; (Equation (10))
      Update trail level τij  for LocalGroupWorst; (Equation (10))
      EndFor
   EndIf
   /*Perform Global Pheromone Update*/
   Compute tour distance of LocalGroupBest
   If (tour distance < GlobalBestTour)
      GlobalBest = LocalGroupBest;
   Else if (tour distance > GlobalWorstTour)
      GlobalWorst = LocalGroupBest;
   Else
      /*Pheromone updates for others*/
      Update trail level τij ; (Equation (11))
   EndIf
   /*Pheromone updates for GlobalBest & GlobalWorst*/
   Update trail level τij  for GlobalBest; (Equation (11))
   Update trail level τij  for GlobalWorst; (Equation (11))
While (until all termination statements satisfied)
```

Fig 4: DACS3 algorithm

Table 1: DACS3 parameter settings

Parameters	Value
Ants population size	10.0
q_0	0.9
β	2.0
p	0.1
Max iterations	10000.0

RESULTS

Table 2 shows the experimental results of DACS3 compared to several other algorithms. Figure 5 shows the percentage difference to optimal distance. Figure 6 shows the time taken to reach final solution. Figure 7a-d shows log graphs of shortest distance versus iterations. The comparison is done for DACS3 with ACS and DACS for datasets Burma14, Oliver30, Berlin52 and KroC100.

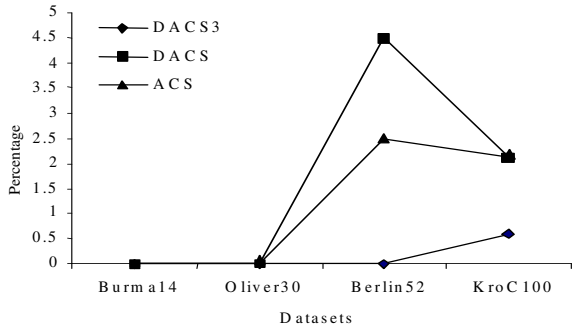

Fig. 5: Percentages differences of shortest distance compare to standard benchmark

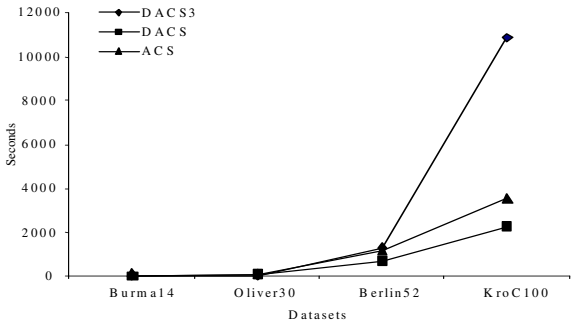

Fig. 6: The performance time taken to reach the shortest distance

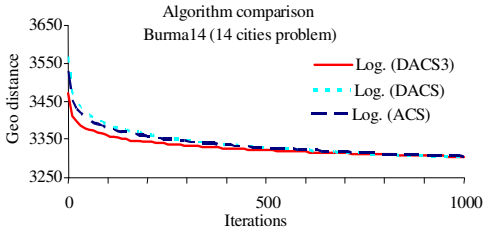

Fig. 7a: Log graph algorithm comparison for Burma14 problem

Table 2: Result comparison between algorithms

	DACS3				DACS			ACS		
TSP Problem	Optimal Distance	Best Distance	Best Iteration	Average Time (sec)	Best Distance	Best Iteration	Average Time (sec)	Best Distance	Best Iteration	Average Time (sec)
Burma14	3323	3323.00	94	1.003	3323.00	527	4.823	3323.00	527	
(14-city problem)	(N/A)	-3330.61			-3330.61			-3330.61		4.767
Oliver30	420	420.00	168	7.043	420.00	1826	71.582	420.00	1826	
(30-city problem)	-423.74	-423.74			-423.74			-423.74		72.501
Berlin52	7542	7542.00	6447	1278.590	7881.00	3661	694.835	7732.00	6050	
(52-city problem)	(N/A)	-7544.37			-7880.78			-7734.11		1165.081
KroC100	20749	20880.00	7954	10856.385	21190.00	1720	2242.229	21170.00	2670	
(100-city problem)	(N/A)	-20881.61			-21191.37			-21172.26		3532.969

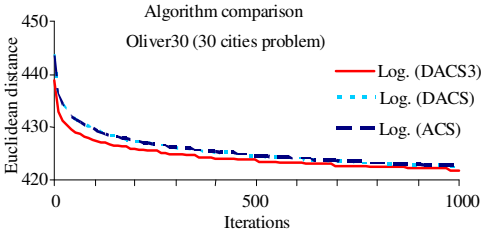

Fig. 7b: Log graph algorithm comparison for Oliver30 problem

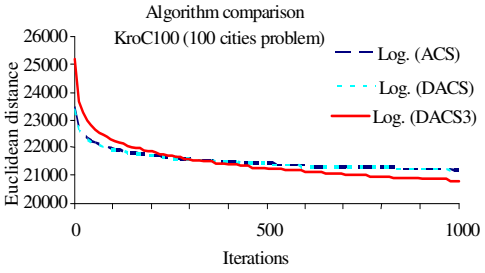

Fig. 7d: Log graph algorithm comparison for KroC100 problem

DISCUSSION

From the result, DACS3 reached the optimal benchmark distance for all case studies except for KroC100, where the difference is very small (0.6%). ACS and DACS are also able to reach the optimal benchmark distance for the Oliver30 and Burma14 data. ACS takes 2.5 and 2.0% longer and DACS takes 4.4 and 2.1% longer for the Berlin52 and KroC100 datasets respectively. In term of time to solution, DACS3 performs 90 and 75% faster compared to ACS and DACS for Oliver30 and Burma14 respectively. However, DACS3 takes a longer time to reach the final solution for larger datasets (Berlin52 and KroC100) but DACS3 achieved better solution quality compared to ACS and DACS.

DACS3 has obtained good searching performance on small-sized problems (Burma 14 and Oliver 30). This is shown by the fact that DACS3 has outperforms other algorithms throughout the run. However, when DACS3 searches for solutions for slightly bigger problems (Berlin50 and KroC100), it performs poorly at the beginning of the search but produces good performance in the middle and end of the run.

CONCLUSION

Based on the results stated above, we can concludes that manipulating and empowering the available knowledge of the individual ants can provide a significant advantage in solving the problem as a whole. Harnessing the experiences of every single individual can expedite the process of finding a good or better solution, either in shorter distance or time to solution. We conclude that DACS3 has outstanding search performance for smaller datasets, but performs slightly worse at the beginning of the search on larger datasets. Since DACS3 has reach shortest distance among other, therefore, the next step is to optimize the search performance of DACS3 using various strategic techniques such as candidate list, Pheromone Trails Smoothing (PTS) and the elitist ant concepts.

REFERENCES

1. Dorigo, M., G.D. Caro and L.M. Gambardella, 1999. Ant algorithms for discrete optimization. J. Artifi. Life, 5: 137-172. DOI: 10.1162/106454699568728.
2. Fleischer, M., 2003. Foundation of swarm intelligence: From principles to practice. http://arxiv.org/PS_cache/nlin/pdf/0502/0502003v1.pdf.
3. Dorigo, M., V. Maniezzo and A. Colorni, 1996. The ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybernet. Part B. Cybernet., 26: 29-41. DOI: 10.1109/3477.484436.

4. Bonabeau, E. and C. Meyer, 2001. Swarm Intelligence: A whole new way to think about business. Harvard Bus. Rev., 79: 107-114. http://www.antoptima.com/admin/pdfrassegna2/pdf028.pdf.
5. Grosan, C. and A. Abraham, 2006. Stigmergic optimization: Inspiration, technologies and perspectives. Stud. Comput. Intel., 31: 1-24. http://www.springerlink.com/content/q421653021119050/.
6. Dorigo, M. and G.D. Caro, 1999. The Ant Colony Optimization Meta-Heuristic. In: New Ideas in Optimization, Fred Glover, Marco Dorigo and David Corne (Eds.). McGraw-Hill, New York, USA., ISBN: 0-07-709506-5, pp: 11-32.
7. Deneubourg, J.L., S. Aron, S. Goss and J.M. Pasteels, 1990. The self-organizing exploratory pattern of argentine ant. J. Insect Behav., 3: 159-168. DOI: 10.1007/BF01417909.
8. Dorigo, M., E. Bonabeau and G. Theraulaz, 2000. Ant algorithms and stigmergy. J. Future Generat. Comput. Syst., 16: 851-871. DOI: 10.1016/S0167-739X(00)00042-X.
9. Anderson, C. and N.R. Franks, 2001, Teams in animal societies, J. Behav. Ecol., 12: 534-540. http://beheco.oxfordjournals.org/cgi/content/abstract/12/5/534.
10. Rais, H.M., Z.A. Othman and A.R. Hamdan, 2007. Improved Dynamic Ant Colony System (DACS) on symmetric Traveling Salesman Problem (TSP). International Conference on Intelligent and Advanced System, Nov. 25-28, IEEE Computer Society, Washington DC., USA., pp: 43-48. DOI: 10.1109/ICIAS.2007.4658345.
11. Dorigo, M. and L.M. Gambardella, 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. Evolut. Comput., 1: 53-66. DOI: 10.1109/4235.585892.
12. Gaertner, D. and K. Clark, 2005. On optimal parameters for Ant colony optimization algorithms. Proceedings of International Conference on Artificial Intelligent, June 27-30, CSREA Press, USA., pp: 74-82. http://eprints.lancs.ac.uk/7927/.
13. Gambardella, L.M. and M. Dorigo, 1996, Solving symmetric and asymmetric TSPs by ant colonies. Proceedings of the IEEE International Conference on Evolutionary Computation, May 20-26, IEEE Computer Society, USA., pp: 622-627. DOI: 10.1109/ICEC.1996.542672.
14. Dorigo, M., M. Birattari and T. Stutzle, 2006. Ant colony optimization: Artificial ants as a computational intelligence technique. IEEE Comput. Intel. Mag., 1: 28-39. DOI : 10.1109/MCI.2006.329691.
15. Stützle, T. and H.H. Hoos, 2000. MAX-MIN ant system. J. Future Generat. Comput. Syst., 16: 889-914. http://portal.acm.org/citation.cfm?id=348599.348603.
16. Merloti, P.E., 2004. Optimization algorithms inspired by biological ants and swarm behavior. http://www.merlotti.com/EngHome/Computing/AntsSim/AntOptimizationAlg.pdf.
17. Li, Y. and S. Gong, 2003. Dynamic ant colony optimization for TSP. Int. J. Adv. Manufact. Technol., 22: 528-533. DOI: 10.1007/s00170-002-1478-9.
18. Gendreau, M. and J.Y. Potvin, 2005. Metaheuristic in combinatorial optimization. Ann. Operat. Res., 140: 189-213. DOI: 10.1007/s10479-005-3971-7.
19. Marsh, L. and C. Onof, 2008. Stigmergic epistemology, stigmergic cognition. Cognit. Syst. Res. J., 9: 136-149, DOI: 10.1016/j.cogsys.2007.06.009.